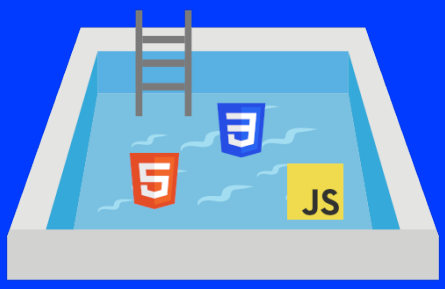
{EPITECH}

WEB DEV POOL_

< DAY 07 - JAVASCRIPT />



WEB DEV POOL

The firsts 7 tasks are provided as HTML files, that will be used to correct your exercises. You have to interact with the HTML code given in order to solve the exercises. You have to make sure that your JavaScript code do NOT require to edit the HTML code.

Editing the HTML code is not necessary.
You don't have to submit the provided HTML files in your delivery.

Today, Javascript is much more powerful than when jQuery came out, **17 years ago**.

Of course, using any external library or code copied from the Internet is forbidden.

Task 01

Delivery: `script/task01.js`
Resource: `task01.html`

Display the number of clicks made in the white box block.

Task 02

Delivery: `script/task02.js`
Resource: `task02.html`

Clicking on the white block makes a dialog box appears and prompts ”What's your name?”.
If no name is filled, the dialog must keep showing up until a name is filled.
Once it's filled, a confirmation box appears to display ”Are you sure that **name** is your name ?”
If it's confirmed, display ”Hello **name**!” in an alert box and also in the white box.

Task 03

Delivery: `script/task03.js`
Resource: `task03.html`

In the white box, display the last 42 characters entered from the keyboard on this page.

Task 04

Delivery: `script/task04.js`
Resource: `task04.html`

Make the buttons `+` and `-` respectively increase and decrease the page's font size.

Make the dropdown menu change the page background color.

Task 05

Delivery: `script/task05.js`
Resource: `task05.html`

Draw a white triangle inside the canvas with:

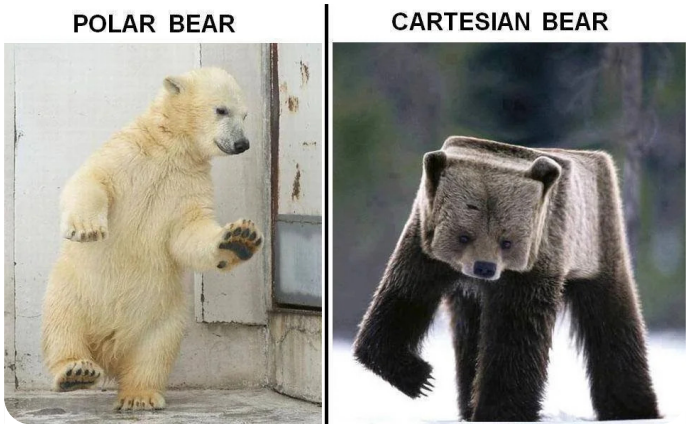
- ✓ a 1 pixel border ;
- ✓ the following vertices' coordinates: {6, 6}, {14, 10}, {6, 14}.

A click on the play button must play the music found at this URL.
Then, make the control buttons (pause, stop, mute) work as expected.

Task 06

Delivery: `script/task06.js`
Resource: `task06.html`

Ensure we can drag the black square inside the white box to move it.
Then, replace the question marks in the second white box with (Cartesian) coordinates in order to display the relative position of the black square.



Task 07

Delivery: `script/task07.js`
Resource: `task07.html`

Clicking the white box link creates the cookie `acceptsCookies`, expiring in 1 day and value `true`.

The white box message is not be displayed if the cookie is already defined and its value is `true`.

However, you will need to make a second white box appear with a button ”Delete the cookie”.

When this button is clicked:

- ✓ the second white box is deleted ;
- ✓ the cookie is deleted ;
- ✓ the first message reappears.

From now on, you will have to write your own HTML along with the corresponding Javascript file.
For each exercise, create a `.js` file and the appropriate `index.html` to show that your script works.

In the following exercises, the instructions must be executed after the page is fully loaded.

If you feel you are doing a bit of magic, fear not!!! That's because you *are* doing magic!
You are the master of all things, a master among masters. **The DOM shall bow before you.**

Task 08

Delivery: `task08/index.html`, `task08/selector.js`

Your JS file contains a function that:

- ✓ selects all elements of hyperlink type that do not have the target=”_blank” attribute ;
- ✓ makes them semi-transparent with 50% opacity.

Task 09

Delivery: `task09/index.html`, `task09/event.js`

Your JS file contains a function that:

- ✓ assigns a ”click” event on the first ”button” element of the page ;
- ✓ makes all paragraphs of the page disappear when the event is triggered.

Task 10

Delivery: `task10/index.html`, `task10/append.js`

In your page, add a text input with the id ”listItem” and a button.

Then, write a function that:

- ✓ is called when the button is clicked ;
- ✓ takes the input's content as an argument ;
- ✓ adds a div after this element, containing the value of the argument.

Task 11

Delivery: `task11/index.html`, `task11/blue.js`

Your JS file contains a function that:

- ✓ adds the ”blue” class to a paragraph when hovering over it ;
- ✓ toggles the ”highlighted” class on a paragraph when clicking it.

Task 12

Delivery: `task12/index.html`, `task12/script.js`, `task12/style.css`

Define a few CSS classes:

- ✓ `note` sets the border of the element to blue ;
- ✓ `email` sets the border of the element to green ;
- ✓ `todo` sets the border of the element to red.

Then, create a form with:

- ✓ a text input ;
- ✓ a dropdown, containing 3 options (”note”, ”email” and ”todo”) ;
- ✓ a button to submit the form.

When submitted, adds the text:

- ✓ to a bullet list displayed under the form ;
- ✓ with the correct css class assigned to it based on the option selected.

Task 13

Delivery: `task13/index.html`, `task13/script.js`, `task13/style.css`

This task is a direct follow-up to the previous one.

Implement a form that will be used to perform searches amongst the created items.
The form should have a dropdown, a ”search” button and a ”reset” button.

You only need to handle search by types, using a dropdown input with three options.
Search by words will be implemented later.

While searching for ”email”, only the corresponding elements remain visible. The others (note and todo) are not deleted. They are shown again when the search is reset. The same goes for the other options.

Task 14

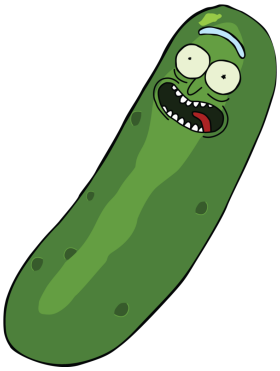
Delivery: `task14/index.html`, `task14/script.js`, `task14/style.css`

Upgrade your search feature to allow an user to search for any string contained in the elements (case insensitively). For instance, searching for ”rick” should display items like:

- ✓ an email from ”rick@sanchez.com” ;
- ✓ a todo ”Trick Patrick and Derrick with some bricks” ;
- ✓ a note ”Awesome list of real pricks” ;
- ✓ ...

Add the possibility to combine search with words and types, such as:

- ✓ note containing ”rick” ;
- ✓ todo containing ”trick” ;
- ✓ email not containing ”rick” ;
- ✓ ...



Task 15

Delivery: `task15/index.html`, `task15/script.js`, `task15/style.css`

This task is a direct follow-up to the previous one.

Finally, make it possible to add tags to an element of the list. The tags must be displayed, and it should be possible to:

- ✓ add multiple tags to the same element, even after its creation ;
- ✓ search for tags ;
- ✓ search for multiple types of elements simultaneously, such as:
  - email and todo containing ”rick” or ”trick” with tag ”urgent”
  - note or todo containing ”rick” with tag ”urgent” and no tag ”done”
  - ...

If you've gotten this far, congratulations! You now have a basic item list with a search engine.
All done in HTML, CSS and Javascript.

v 4.1